CPSC 540: Machine Learning

Mixture Models

Mark Schmidt

University of British Columbia

Winter 2019

Last Time: Mixture of Gaussians

- We discussed density estimation with a mixture of Gaussians,

$$p(x \mid \mu, \Sigma, \pi) = \sum_{c=1}^{k} \pi_c \underbrace{p(x \mid \mu_c, \Sigma_c)}_{\text{PDF of Gaussian } c},$$

  where PDF is written as convex combination of Gaussian PDFs.
  - Convex combination is needed so that probability integrates to 1.

- More flexible than a single Gaussian.
- With enough Gaussians, can approximate any continuous PDF.

- More generally, we can have mixtures of any distributions.
  - Today we'll discuss mixture of Bernoullis.
  - You can also do mixture of student t, mixture of Poisson, and so on.

Previously: Independent vs. General Discrete Distributions

- We previously considered density estimation with discrete variables,

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

  and considered two extreme approaches:
  - Product of independent Bernoullis:

$$p(x^i \mid \theta) = \prod_{j=1}^{d} p(x_j^i \mid \theta_j).$$

    Easy to fit but strong independence assumption:
    - Knowing x_j^i tells you nothing about x_k^i.
  - General discrete distribution:

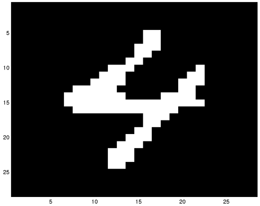
$$p(x^i \mid \theta) = \theta_{x^i}.$$

    No assumptions but hard to fit:
    - Parameter vector θ_{x^i} for each possible x^i.

Independent vs. General Discrete Distributions on Digits

- Consider handwritten images of digits:

$x^i = \text{vec}\Big($



$\Big),$

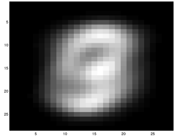
so each row of X contains all pixels from one image of a 0, 1, 2, . . . , or a 9.

- Previously we had labels and wanted to recognize that this is a 4.
- In density estimation we want probability distribution over images of digits.

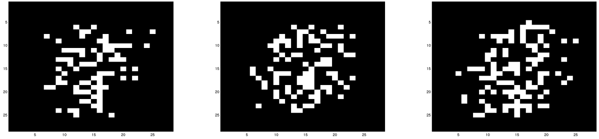
- Given an image, what is the probability that it's a digit?
- Sampling from the density estimator it should generate images of digits.

Independent vs. General Discrete Distributions on Digits

- Fitting independent Bernoullis to this data gives a parameter θ_j for each pixel j.
  - "Fraction of times we have a 1 at pixel j":



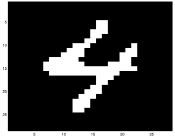
- Samples generated from independent Bernoulli model:



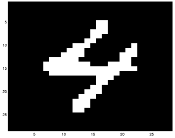
  - Flip a coin that lands hands with probability θ_j for each pixel j.
- This is clearly a terrible model: misses dependencies between pixels.

Independent vs. General Discrete Distributions on Digits

- Here is a sample from the MLE with the general discrete distribution:



- Here is an image with a probability of 0:



- This model memorized training images and doesn't generalize.
  - MLE puts probability at least $1/n$ on training images, and 0 on non-training images.

- A model lying between these extremes is the mixture of Bernoullis.

Mixture of Bernoullis

- Consider a coin flipping scenario where we have two coins:
  - Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (biased).
- Half the time we flip coin 1, and otherwise we flip coin 2:

$$
\begin{aligned}
p(x^i = 1 \mid \theta_1, \theta_2) &= \pi_1 p(x^i = 1 \mid \theta_1) + \pi_2 p(x^i = 1 \mid \theta_2) \\
&= \frac{1}{2}\theta_1 + \frac{1}{2}\theta_2 = \frac{\theta_1 + \theta_2}{2}
\end{aligned}
$$

- With one variable this mixture model is not very interesting:
  - It's equivalent to flipping one coin with $\theta = 0.75$.
- But with multiple variables mixture of Bernoullis can model dependencies...

Mixture of Independent Bernoullis

- Consider a mixture of independent Bernoullis:

$$p(x \mid \theta_1, \theta_2) = \frac{1}{2} \underbrace{\prod_{j=1}^{d} p(x_j \mid \theta_{1j})}_{\text{first set of Bernoullis}} + \frac{1}{2} \underbrace{\prod_{j=1}^{d} p(x_j \mid \theta_{2j})}_{\text{second set of Bernoulli}} .$$

- Conceptually, we now have two sets of coins:
  - Half the time we throw the first set, half the time we throw the second set.
- With $d = 4$ we could have $\theta_1 = \begin{bmatrix} 0 & 0.7 & 1 & 1 \end{bmatrix}$ and $\theta_2 = \begin{bmatrix} 1 & 0.7 & 0.8 & 0 \end{bmatrix}$.
  - Half the time we have $p(x_3^i = 1) = 1$ and half the time it's 0.8.
- Have we gained anything?

Mixture of Independent Bernoullis

- Example from the previous slide: $\theta_1 = \begin{bmatrix} 0 & 0.7 & 1 & 1 \end{bmatrix}$ and $\theta_2 = \begin{bmatrix} 1 & 0.7 & 0.8 & 0 \end{bmatrix}$.
- Here are some samples from this model:

$$X = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

- Unlike product of Bernoullis, notice that features in samples are not independent.
  - In this example knowing $x_1 = 1$ tells you that $x_4 = 0$.
- This model can capture dependencies: $\underbrace{p(x_4 = 1 \mid x_1 = 1)}_{0} \neq \underbrace{p(x_4 = 1)}_{0.5}$.

Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:

$$p(x^i \mid \Theta) = \sum_{c=1}^{k} \pi_c p(x^i \mid \theta_c),$$

  where Θ contains all the model parameters.

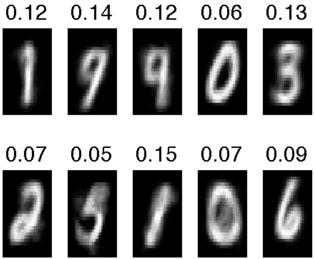
- Mixture of Bernoullis can model dependencies between variables
  - Individual mixtures act like clusters of the binary data.
  - Knowing cluster of one variable gives information about other variables.

- With k large enough, mixture of Bernollis can model any discrete distribution.
  - Hopefully with $k << 2^d$.

Mixture of Independent Bernoullis

- Plotting parameters θ_c with 10 mixtures trained on MNIST digits (with "EM"):

  (hand-written images of the the numbers 0 through 9, numbers above images are mixture coefficients π_c)

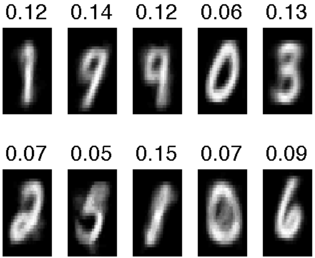


http://pmtk3.googlecode.com/svn/trunk/docs/demoOutput/bookDemos/%2811%29-Mixture_models_and_the_EM_algorithm/mixBerMnistEM.html

- Remember this is unsupervised: it hasn't been told there are ten digits.
  - Density estimation is trying to figure out how the world works.

Mixture of Independent Bernoullis

- Plotting parameters θ_c with 10 mixtures trained on MNIST digits (with "EM"):
  (hand-written images of the the numbers 0 through 9, numbers above images are mixture coefficients π_c)



http://pmtk3.googlecode.com/svn/trunk/docs/demoOutput/bookDemos/%2811%29-Mixture_models_and_the_EM_algorithm/mixBerMnistEM.html

- You could use this model to "fill in" missing parts of an image:
  - By finding likely cluster/mixture, you find likely values for the missing parts.

Generative Classifiers: Supervised Learning with Density Estimation

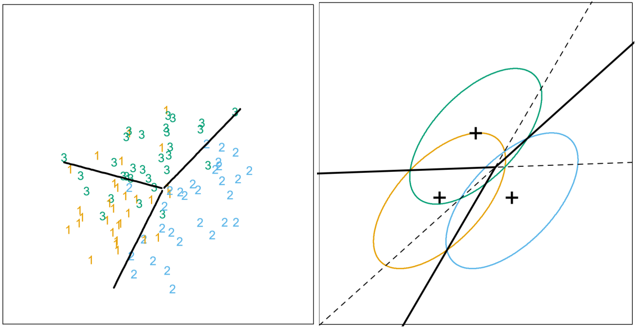
- Density estimation can be used for supervised learning:
  - Generative classifiers estimate conditional by modeling joint probability of x^i and y^i,

$$\begin{aligned} p(y^i \mid x^i) &\propto p(x^i, y^i) && \text{(Approach 1: model joint probability of } x^i \text{ and } y^i\text{)} \\ &= p(x^i \mid y^i)p(y^i). && \text{(Approach 2: model marginal of } y^i \text{ and conditional)} \end{aligned}$$

- Common generative classifiers (based on Approach 2):
  - Naive Bayes models $p(x^i \mid y^i)$ as product of independent distributions.
    - Has recently been used for CRISPR gene editing.
  - Linear discriminant analysis (LDA) assumes $p(x^i \mid y^i)$ is Gaussian (shared Σ).
  - Gaussian discriminant analysis (GDA) allows each class to have its own covariance.

Linear Discriminant Analysis (LDA)

- Example of fitting linear discriminant analysis (LDA) to a 3-class problem:

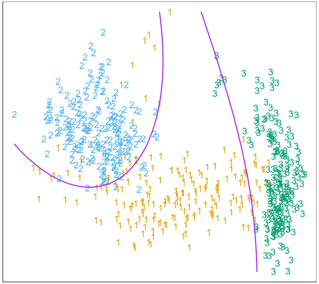


https://web.stanford.edu/~hastie/Papers/ESLII.pdf

- Gaussian for each class with same Σ leads to a linear classifier.
  - Class label is determined by nearest mean.

Gaussian Discriminant Analysis (GDA)

- Example of fitting Gaussian discriminant analysis (GDA) to a 3-class problem:



https://web.stanford.edu/~hastie/Papers/ESLII.pdf

- Different Σ_c for each class c leads to a quadratic classifier.
  - Class label is determined by means and variances.

Digression: Generative Models for Structured Prediction

- Consider a structured prediction problem where target y^i is a vector:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

- Approach 2 (modeling $x^i \mid y^i$) leas to too many y^i potential values.
- But you could model joint probability of x^i and y^i (Approach 1),

$$Z = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

- So any of the density estimation we discuss can be used.
  - Given $p(x^i, y^i)$ use conditioning to get $p(y^i \mid x^i)$ to make predictions.

Beyond Naive Bayes and GDA

- GDA and naive Bayes make strong assumptions.
  - That features x^i are independent or Gaussian (respectively) given labels y^i.
- You can get a better model of each class by using a mixture model for $p(x^i \mid y^i)$.
- Generative models were unpopular for a while, but are coming back:
  - Generative adversarial networks (GANs) and variational autoencoders.
    - Deep generative models (later in course).
  - We believe that most human learning is unsupervised.
    - There may not be enough information in class labels to learn quickly.
    - Instead of searching for features that indicate "dog", try to model all aspects of dogs.

Outline

Gaussian Discriminant Analysis (GDA) and Closed-Form MLE

- In Gaussian discriminant analysis we assume $x^i \mid y^i$ is a Gaussian.

$$p(x^i, y^i = c) = \underbrace{\pi_c}_{p(y^i=c)} \underbrace{p(x^i \mid \mu_c, \Sigma_c)}_{\text{Gaussian PDF}}.$$

- If we don't know y^i, this is actually a mixture of Gaussians model:

$$p(x^i) = \sum_{c=1}^{k} p(x^i, y^i = c) = \sum_{c=1}^{k} \pi_c p(x^i \mid \mu_c, \Sigma_c).$$

- But since we know which "cluster" each x^i comes from, MLE is simple:

$$\hat{\pi}_c = \frac{n_c}{n}, \quad \hat{\mu}_c = \frac{1}{n_c}\sum_{y^i=c} x^i, \quad \hat{\Sigma}_c = \frac{1}{n_c}\sum_{y^i=c}(x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T,$$

"use the sample statistics for examples in class c".

- Methods for fitting mixtures models treat "clusters" as hidden values.

Learning with Hidden Values

- We often want to learn with unobserved/missing/hidden/latent values.
- For example, we could have a dataset like this:

$$X = \begin{bmatrix} N & 33 & 5 \\ L & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Missing values are very common in real datasets.
- An important issue to consider: why is data missing?

Missing at Random (MAR)

- We'll focus on data that is missing at random (MAR):
  - Assume that the reason ? is missing does not depend on the missing value.
    - Formal definition in bonus slides.
  - This definition doesn't agree with intuitive notion of "random":
    - A variable that is *always* missing would be "missing at random".
    - The intuitive/stronger version is missing completely at random (MCAR).
- Examples of MCAR and MAR for digit data:
  - Missing random pixels/labels: MCAR.
  - Hide the the top half of every digit: MAR.
  - Hide the labels of all the "2" examples: not MAR.
- We'll consider MAR, because otherwise you need to model why data is missing.

Imputation Approach to MAR Variables

- Consider a dataset with MAR values:

$$X = \begin{bmatrix} N & 33 & 5 \\ F & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Imputation method is one of the first things we might try:
  0. Initialization: find parameters of a density model (often using "complete" examples).
  1. Imputation: replace each ? with the most likely value.
  2. Estimation: fit model with these imputed values.

- You could also alternate between imputation and estimation.
  - Block coordinate optimization, treating ? values as more parameters.

Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

$$X = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}, \quad y = \begin{bmatrix} \\ \\ \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} & & \\ & & \\ & & \\ & & \\ & & \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}.$$

- Motivation for training on labeled data (X, y) and unlabeled data $\bar{X}$:
  - Getting labeled data is usually expensive, but unlabeled data is usually cheap.

Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

$$X = \begin{bmatrix} \\ \\ \\ \end{bmatrix}, \quad y = \begin{bmatrix} \\ \\ \\ \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix},$$

- Imputation approach is called self-taught learning:
  - Alternate between guessing $\bar{y}$ and fitting the model with these values.

Back to Mixture Models

- To fit mixture models we often introduce n MAR variables z^i.
- Why???
- Consider mixture of Gaussians, and let z^i be the cluster number of example i:
  - So $z^i \in \{1, 2, \cdots, k\}$ tells you which Gaussian generated example i.
  - Given $\{\pi_c, \mu_c, \Sigma_c\}$ it's easy to optimize the clusters z^i:
    - Find the cluster c maximizing $p(x^i, z_i = c)$ (prediction step in GDA).
  - Given the z^i it's easy to optimize the parameters of the mixture model.
    - Solve for $\{\pi_c, \mu_c, \Sigma_c\}$ maximizing $p(x^i, z^i)$ (learning step in GDA).

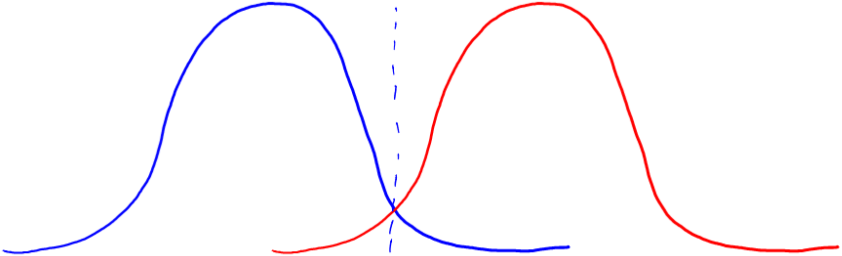
Imputation Approach for Mixtures of Gaussians

- Consider mixture of Gaussians with the choice $\pi_c = 1/k$ and $\Sigma_c = I$ for all c.
- Here is the imputation approach for fitting a mixtures of Gaussian:
  - Randomly pick some initial means μ_c.
  - Assigns x^i to the closest mean..
    - This is how you maximize $p(x^i, z^i)$ in terms of z^i.
  - Set μ_c to the mean of the points assigned to cluster c.
    - This is how you maximize $p(x^i, z^i)$ in terms of μ_c.
- This is exactly k-means clustering.

K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.

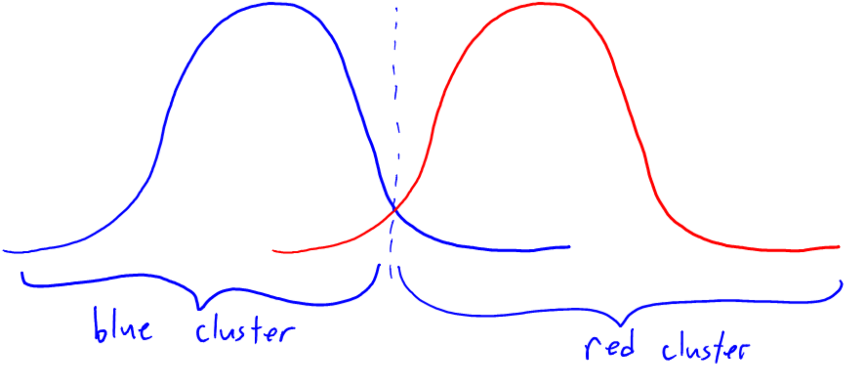
With same covariance, clusters are convex.



K-Means vs. Mixture of Gaussians

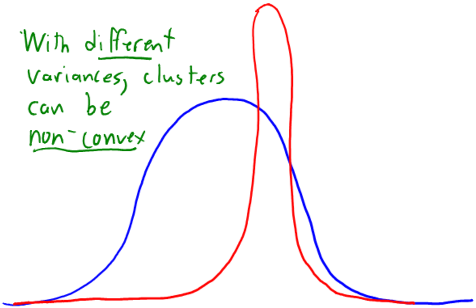
- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.

With same covariance, clusters are convex.



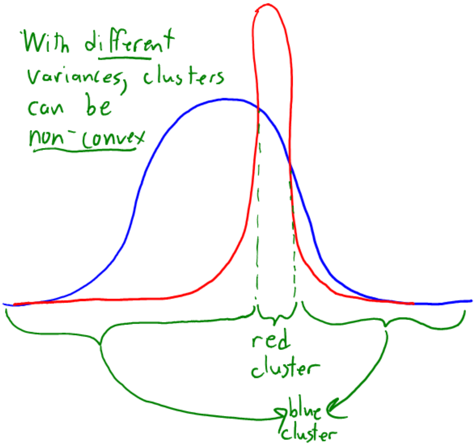
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



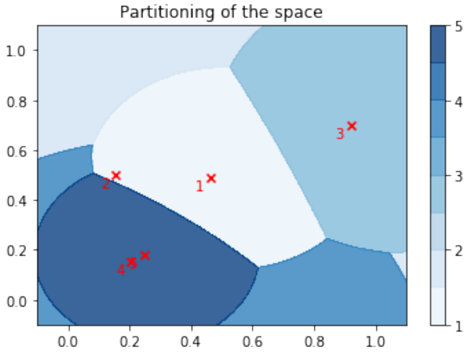
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



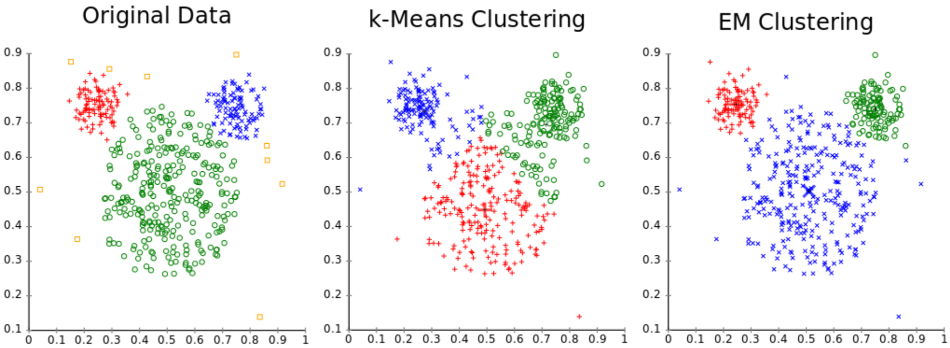
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



https://en.wikipedia.org/wiki/K-means_clustering

Drawbacks of Imputation Approach

- The imputation approach to MAR variables is simple:
  - Use density estimator to "fill in" the missing values.
  - Now fit the "complete data" using a standard method.
- But "hard" assignments of missing values lead to propagation of errors.
  - What if cluster is ambiguous in k-means clustering?
  - What if label is ambiguous in "self-taught" learning?
- Ideally, we should use probabilities of different assignments ("soft" assignments):
  - If the MAR values are obvious, this will act like the imputation approach.
  - For ambiguous examples, takes into account probability of different assignments.
- Expectation maximization (EM) considers probability of all imputations of ?.

Summary

- Mixture of Bernoullis can model dependencies between discrete variables.
  - Probability of belonging to mixtures is a soft-clustering of examples.
- Generative classifiers turn supervised learning into density estimation.
  - Naive Bayes and GDA are popular, but make strong assumptions.
  - Can be used for structured prediction.
- Missing at random: fact that variable is missing does not depend on its value.
- Imputation approach to handling missing data.
  - Guess values of hidden variables, then fit the model (and usually repeat).
  - K-means is a special case, if we introduce "cluster number" as MAR variables.
- Next time: one of the most cited papers in statistics.

Missing at Random (MAR) Formally

- Let's formally define MAR in the context of density estimation.
- Our "observed" data would be a matrix X containing ? values.
- Our "complete" data would be the matrix X the ? values "filled in".
  - Let x_j^i be the value in this matrix, which may be a ? in the observed data.
- Use $z_j^i = 1$ if x_i^j is ? in the "observed" data.
- We say that data is MAR in the observed data X if

$$z_j^i \perp x_j^i,$$

that the fact that x_j^i is missing (z_j^i) is independent of the value of x_j^i.
  - Specific values of the variables are not being hidden.